

# **Trabajo Práctico 0: Infraestructura básica**

Fabrizio Cozza, *Padrón Nro. 97.402*  
fabrizio.cozza@gmail.com

Kevin Cajachuán, *Padrón Nro. 98.725*  
kevincajachuan@hotmail.com

Luciano Giannotti, *Padrón Nro. 97.215*  
luciano\_giannotti@hotmail.com.ar

1er. Cuatrimestre de 2018  
66.20 Organización de Computadoras – Práctica Martes  
Facultad de Ingeniería, Universidad de Buenos Aires

## 1. Objetivos

Este Trabajo Práctico tiene el fin de ayudarnos a familiarizarnos con las herramientas de software que utilizaremos posteriormente en otros trabajos, como es el emulador **gxemul** para correr programas sobre una maquina MIPS con el Sistema Operativo NetBSD.

## 2. Programa

El software de este trabajo esta escrito en lenguaje C y permite dibujar **Julia Sets** o **Conjuntos de Julia** segun los parámetros que le pasamos por línea de comando. Estos parámetros son la region del plano complejo: delimitada por un centro, un ancho y un alto; una semilla que afectara el calculo para cada pixel; la resolución y la salida ya sea por pantalla o por archivo. El formato a usar es PGM o *portable gray format*, que resulta útil para describir imágenes digitales en escala de grises.

## 3. Implementación

Una vez recibidos los parámetros, para dibujar el Julia Set el programa obtiene de cada píxel de la ventana a un punto en el plano complejo. A ese punto se lo eleva al cuadrado y le suma la semilla mencionada en la sección anterior. Esto se repite hasta que el valor absoluto del resultado sea menor a 2, en cuyo caso se toma la cantidad de iteraciones y se imprime en el archivo PGM, representando el nivel de blanco de ese píxel.

## 4. Código C

En esta sección colocaremos el código fuente del programa en lenguaje C.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#define DEFAULT_WIDTH_RES 640;
#define DEFAULT_HEIGHT_RES 480;
#define DEFAULT_REALIMAGINARY 0;
#define DEFAULT_WIDTH_HEIGHT 2;
#define DEFAULT_REALSEED -0.726895347709114071439;
#define DEFAULT_IMAGINARYSEED 0.188887129043845954792;

typedef struct{
    double x,y;
}complex;

complex addComplexNumbers(complex a,complex b){
    complex c;
    c.x = a.x + b.x;
    c.y = a.y + b.y;
    return c;
}

complex sqrComplex(complex a){
    complex c;
    c.x = a.x*a.x - a.y*a.y;
    c.y = 2*a.x*a.y;
    return c;
}
```

```

}

double absComplex(complex a){
    return sqrt(a.x*a.x + a.y*a.y);
}

int processImage(int resW, int resH,
                complex pPos, complex seed,
                double w, double h,
                FILE* im, int N){

    int x,y,i;
    int data[resH][resW];
    complex z0,z1;

    for(y=0;y<resH;y++){
        for(x=0;x<resW;x++){
            // Valor z segun posicion del pixel
            z1.x=pPos.x-w/2+w/(double)resW/2+w/(double)resW*x;
            z1.y=pPos.y+h/2-h/(double)resH/2-h/(double)resH*y;
            z0.x = 0;
            z0.y = 0;
            for(i=0;i<N-1;i++){
                z0 = addComplexNumbers(sqrComplex(z1),seed);
                z1=z0;
                if (absComplex(z0) > 2.0){
                    break;
                }
                i++;
            }
            //agregar al buffer el brillo
            data[y][x] = i;
        }
    }

    //Empezar a imprimir el pgm
    fprintf(im, "P2 \n");
    fprintf(im, "%d %d \n",resW,resH);
    fprintf(im, "%d \n", N);
    y = 0;
    while(y < resH)
    {
        x = 0;
        while (x < resW)
        {
            fprintf(im, "%3d ", data[y][x]);
            x++;
        }
        fprintf(im, "\n");
        y++;
    }

    /* close the file */
    return fclose(im);
}

int main(int argc, char* argv[])
{
    int exitCode = 0;

    int resWidth;
    int resHeight;
    complex pixelPos;
    double width;
    double height;
    complex seed;

    const char delimitator[4] = "x+-i";
    char *pSeparator;

    resWidth = DEFAULT_WIDTH_RES;
    resHeight = DEFAULT_HEIGHT_RES;
    pixelPos.x = DEFAULT_REALIMAGINARY;

```

```

    pixelPos.y = DEFAULT_REALIMAGINARY;
    width = DEFAULT_WIDTH_HEIGHT;
    height = DEFAULT_WIDTH_HEIGHT;
    seed.x = DEFAULT_REALSEED;
    seed.y = DEFAULT_IMAGINARYSEED;

    for (int i = 1; i < argc; ++i){
        if (((!strcmp(argv[i], "-V")) || (!strcmp(argv[i], "--version")))){
            printf("TP0 Organizacion de Computadoras version \"1.0.0\"\\n\\nIntegrantes:\\n Fabrizio Cozza\\n Kevin Cajachua\\n\\nLuciano Giannotti\\n");
            return 0;
        }
        if (((!strcmp(argv[i], "-h")) || (!strcmp(argv[i], "--help")))){
            printf("\\n
    Uso:\\n\\n
    tp0 -h\\n\\n
    tp0 -V\\n\\n
    tp0 [options]\\n\\n
    Opciones:\\n\\n
    -V, --version      Version del programa.\\n\\n
    -h, --help         Informacion acerca de los comandos.\\n\\n
    -r, --resolution   Cambiar resolucion de la imagen.\\n\\n
    -c, --center        Coordenadas correspondientes al punto central.\\n\\n
    -w, --width         Especifica el ancho de la region del plano complejo por
                        dibujar.\\n\\n
    -H, --height        Especifica el alto de la region del plano complejo por
                        dibujar.\\n\\n
    -s, --seed          Configurar el valor complejo de la semilla usada para
                        generar el fractal.\\n\\n
    -o, --output        Colocar la imagen de salida.\\n\\n
    Ejemplos:\\n\\n
    tp0 -o uno.pgm\\n");
            return 0;
        }

        if (!strcmp(argv[i], "-r") ||
            !strcmp(argv[i], "--resolution")){
            if(!argv[i+1]){
                printf("Error: valor de resolucion ingresado no valido\\n
            ");
                return -1;
            }
            else {
                pSeparator = strtok(argv[i+1], delimiter);
                if(pSeparator != NULL){
                    resWidth = atof(pSeparator);
                    if(resWidth <= 0){
                        exitCode = -2;
                    }
                }
                else {
                    exitCode = -1;
                }
            }

            pSeparator = strtok(NULL, delimiter);
            if(pSeparator != NULL){
                resHeight = atof(pSeparator);
                if(resHeight <= 0){
                    exitCode = -2;
                }
            }
            else {
                exitCode = -1;
            }
        }

        if (!strcmp(argv[i], "-c") ||
            !strcmp(argv[i], "--center")){
            if(!argv[i+1]){
                printf("Error: valor de centro ingresado no valido\\n");
                return -1;
            }
            else {
                char *copy = strdup(argv[i+1]);

```

```

        if(copy == NULL){
            exitCode = -1;
        }

        int sign = 1;
        if(copy[0] == '-') sign = -1;

        pSeparator = strtok(argv[i+1], delimiter);
        if(pSeparator != NULL){
            pixelPos.x = sign * atof(pSeparator);
            int len = strlen(pSeparator);
            if(sign == -1) sign = copy[
                len + 1] == '-' ? -1 :
                1;
            else sign = copy[len] == '-'
                ? -1 : 1;
        } else {
            exitCode = -1;
        }

        pSeparator = strtok (NULL, delimiter);
        if(pSeparator != NULL){
            pixelPos.y = sign * atof(pSeparator);
        } else {
            exitCode = -1;
        }

        free(copy);
    }

    if (!strcmp(argv[i], "-w") ||
        !strcmp(argv[i], "--width")){
        if(!argv[i+1]){
            printf("Error: valor de ancho ingresado no valido\n");
            return -1;
        } else {
            width = atof(argv[i+1]);
        }
    }

    if (!strcmp(argv[i], "-H") ||
        !strcmp(argv[i], "--height")){
        if(!argv[i+1]){
            printf("Error: valor de altura ingresado no valido\n");
            return -1;
        } else {
            height = atof(argv[i+1]);
        }
    }

    if (!strcmp(argv[i], "-s") ||
        !strcmp(argv[i], "--seed")){
        if(!argv[i+1]){
            printf("Error: valor de seed ingresado no valido\n");
            return -1;
        } else {
            char *copy = strdup(argv[i+1]);
            if(copy == NULL){
                exitCode = -1;
            }

            int sign = 1;
            if(copy[0] == '-') sign = -1;

            pSeparator = strtok(argv[i+1],
                delimiter);
            if(pSeparator != NULL){
                seed.x = sign * atof(pSeparator);
                int len = strlen(pSeparator);
                if(sign == -1) sign = copy[
                    len + 1] == '-' ? -1 :
                    1;
                else sign = copy[len] == '-'
                    ? -1 : 1;
            } else {
                exitCode = -1;
            }
        }
    }

```

```

    }

    pSeparator = strtok (NULL,delimiter
    );
    if(pSeparator != NULL){
        seed.y = sign * atof(pSeparator);
    } else {
        exitCode = -1;
    }

    free(copy);
}

if (!strcmp(argv[i], "-o") ||
    !strcmp(argv[i], "--output")){

    /* open output file */

    FILE* image;
    int pasoN = 500;
    if (!strcmp(argv[i
        +1], "-")){
        image
        =
        stdout
        ;

    }

    else {
        image = fopen(argv[i+1], "w");
        if (image == NULL)
        {
            fprintf(stderr, "No se puede abrir el archivo file
                %s!\n", argv[i+1]);
            return -1;
        }

    }

    if(exitCode == 0){
        exitCode = processImage(resWidth,resHeight,
            pixelPos,seed,width,height,image,pasoN);
    }

}

switch (exitCode){
case 0:
    return exitCode;
    break;
case -1:
    fprintf(stderr, "La imagen no se pudo procesar, por favor revise los
        valores ingresados\n");
    return exitCode;
    break;
case -2:
    fprintf(stderr, "Valores ingresados de resolucion invalidos\n");
    return exitCode;
    break;
}
return 0;
}

```

## 5. Pruebas

Para las pruebas compilamos el programa con gcc de la siguiente manera:

```
$gcc main.c -o tp0
```

Luego corremos el archivo **test.sh**. Ya que las pruebas son sobre las imágenes, las vamos a realizar a ojo comparandolas con las del enunciado y con las obtenidas en un generador online (<http://usefuljs.net/fractals/>).

Cabe destacar que las imágenes del generador tienen mayor rango dinámico que las del enunciado y nosotros decidimos generarlas como en éste último.

Las imágenes obtenidas por nuestro trabajo se encuentran también en formato PNG en la subcarpeta *imagenes*. A su vez las imágenes del generador online se encuentran en *Casos de prueba*.

### 5.1. Caso con los valores por defecto

Se obtiene una imagen como la primera figura del enunciado:

```
$/tp0 -o uno.pgm
```

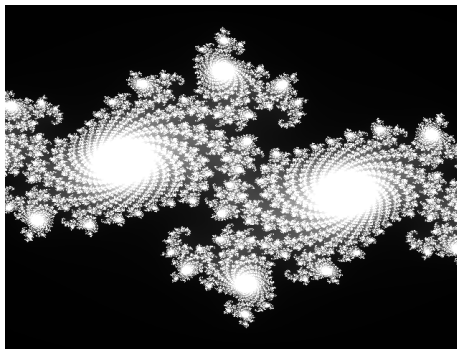


Figura 1:

## 5.2. Caso de imagen con zoom y otro centro

Se obtiene una imagen como la segunda figura del enunciado:

```
$ ./tp0 -c 0.282-0.007i -w 0.005 -H 0.005 -o dos.pgm
```

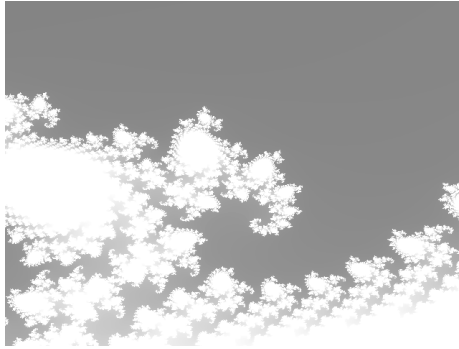


Figura 2:

## 5.3. Caso de imagen con ancho 1 y centro 1

Se obtiene una imagen como la primera del enunciado pero con un zoom x2 aplicado:

```
$ ./tp0 -w 1 -H 1 -o tres.pgm
```

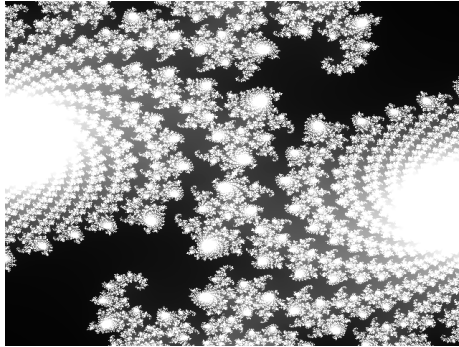


Figura 3:



#### 5.4. Caso de imagen muy chica

Imprimimos una imagen de 8x6 para que se puedan notar claramente los pixeles en la pantalla.

```
$ ./tp0 -r 8x6 -o cuatro.pgm
```

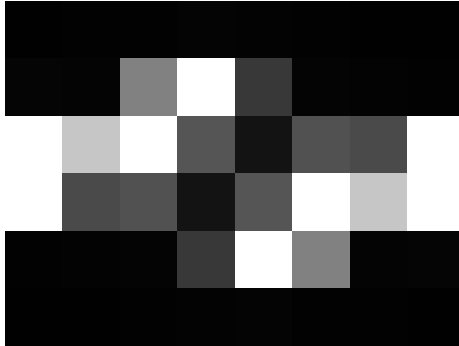


Figura 4:

#### 5.5. Caso de imagen con otra semilla

Esta imagen usa una semilla con sus dos componentes negativas y la imaginaria mucho mas grande que la real.

```
$ ./tp0 -s -0.157-1.041i -o cinco.pgm
```

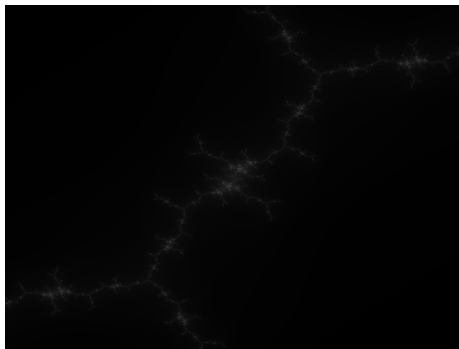


Figura 5:

## 5.6. Caso de imagen muy angosta

En este caso cambiamos la resolución para obtener una imagen de un pixel de alto y 800 de ancho.

Es difícil observarla en el informe por lo que decidimos no colocarla. Sin embargo el archivo se encuentra en la misma carpeta del TP con el nombre *seis.pgm*.

```
$ ./tp0 -r 800x1 -o seis.pgm
```

## 6. Código S

En esta sección colocaremos el código assembly MIPS generado por NetBSD.

```
.file 1 "main.c"
.section .mdebug.abi32
.previous
.abicalls
.text
.align 2
.globl addComplexNumbers
.ent addComplexNumbers
addComplexNumbers:
.frame $fp,32,$ra # vars= 16, regs= 2/0, args= 0, extra
= 8
.mask 0x50000000,-4
.fmask 0x00000000,0
.set noreorder
.cpload $t9
.set reorder
subu $sp,$sp,32
.cprestore 0
sw $fp,28($sp)
sw $gp,24($sp)
move $fp,$sp
move $v1,$a0
sw $a2,40($fp)
sw $a3,44($fp)
l.d $f2,40($fp)
l.d $f0,56($fp)
add.d $f0,$f2,$f0
s.d $f0,8($fp)
l.d $f2,48($fp)
l.d $f0,64($fp)
add.d $f0,$f2,$f0
s.d $f0,16($fp)
lw $v0,8($fp)
sw $v0,0($v1)
lw $v0,12($fp)
sw $v0,4($v1)
lw $v0,16($fp)
sw $v0,8($v1)
lw $v0,20($fp)
sw $v0,12($v1)
move $v0,$v1
move $sp,$fp
lw $fp,28($sp)
addu $sp,$sp,32
j $ra
.end addComplexNumbers
.size addComplexNumbers,.-addComplexNumbers
.align 2
.globl sqrComplex
.ent sqrComplex
sqrComplex:
.frame $fp,32,$ra # vars= 16, regs= 2/0, args= 0, extra
= 8
.mask 0x50000000,-4
.fmask 0x00000000,0
```

```

.set      noreorder
.cpload  $t9
.set      reorder
subu      $sp,$sp,32
.cprestore 0
sw        $fp,28($sp)
sw        $gp,24($sp)
move      $fp,$sp
move      $v1,$a0
sw        $a2,40($fp)
sw        $a3,44($fp)
l.d       $f2,40($fp)
l.d       $f0,40($fp)
mul.d     $f4,$f2,$f0
l.d       $f2,48($fp)
l.d       $f0,48($fp)
mul.d     $f0,$f2,$f0
sub.d     $f0,$f4,$f0
s.d       $f0,8($fp)
l.d       $f0,40($fp)
add.d     $f2,$f0,$f0
l.d       $f0,48($fp)
mul.d     $f0,$f2,$f0
s.d       $f0,16($fp)
lw        $v0,8($fp)
sw        $v0,0($v1)
lw        $v0,12($fp)
sw        $v0,4($v1)
lw        $v0,16($fp)
sw        $v0,8($v1)
lw        $v0,20($fp)
sw        $v0,12($v1)
move      $v0,$v1
move      $sp,$fp
lw        $fp,28($sp)
addu      $sp,$sp,32
j         $ra
.end      sqrComplex
.size     sqrComplex, .-sqrComplex
.align    2
.globl    absComplex
.ent      absComplex
absComplex:
.frame    $fp,40,$ra          # vars= 0, regs= 3/0, args= 16, extra
        = 8
.mask     0xd0000000,-8
.fmask    0x00000000,0
.set      noreorder
.cpload  $t9
.set      reorder
subu      $sp,$sp,40
.cprestore 16
sw        $ra,32($sp)
sw        $fp,28($sp)
sw        $gp,24($sp)
move      $fp,$sp
sw        $a0,40($fp)
sw        $a1,44($fp)
sw        $a2,48($fp)
sw        $a3,52($fp)
l.d       $f2,40($fp)
l.d       $f0,40($fp)
mul.d     $f4,$f2,$f0
l.d       $f2,48($fp)
l.d       $f0,48($fp)
mul.d     $f0,$f2,$f0
add.d     $f0,$f4,$f0
mov.d     $f12,$f0
la        $t9,sqrt
jal       $ra,$t9
move      $sp,$fp
lw        $ra,32($sp)
lw        $fp,28($sp)
addu      $sp,$sp,40

```

```

        j          $ra
        .end       absComplex
        .size      absComplex, .-absComplex
        .rdata
        .align     2
$LC1:
        .ascii    "P2 \n\000"
        .align     2
$LC2:
        .ascii    "%d %d \n\000"
        .align     2
$LC3:
        .ascii    "%d \n\000"
        .align     2
$LC4:
        .ascii    "%3d \000"
        .align     2
$LC5:
        .ascii    "\n\000"
        .align     3
$LC0:
        .word      0
        .word      1073741824
        .text
        .align     2
        .globl     processImage
        .ent        processImage
processImage:
        .frame      $fp,144,$ra          # vars= 80, regs= 4/0, args= 40,
                extra= 8
        .mask       0xd0010000,-4
        .fmask      0x00000000,0
        .set        noreorder
        .cpload     $t9
        .set        reorder
        subu        $sp,$sp,144
        .cprestore  40
        sw          $ra,140($sp)
        sw          $fp,136($sp)
        sw          $gp,132($sp)
        sw          $s0,128($sp)
        move        $fp,$sp
        sw          $a0,144($fp)
        sw          $a1,148($fp)
        sw          $a2,152($fp)
        sw          $a3,156($fp)
        sw          $sp,112($fp)
        lw          $v0,148($fp)
        addu        $v1,$v0,-1
        lw          $v0,144($fp)
        addu        $v0,$v0,-1
        sw          $v0,116($fp)
        lw          $v0,116($fp)
        sll         $v0,$v0,2
        addu        $v0,$v0,4
        addu        $v1,$v1,1
        mult        $v0,$v1
        mflo        $v0
        addu        $v0,$v0,7
        srl         $v0,$v0,3
        sll         $v0,$v0,3
        subu        $sp,$sp,$v0
        addu        $v0,$sp,40
        sw          $v0,120($fp)
        sw          $zero,52($fp)
$L21:
        lw          $v0,52($fp)
        lw          $v1,148($fp)
        slt         $v0,$v0,$v1
        bne        $v0,$zero,$L24
        b          $L22
$L24:
        sw          $zero,48($fp)
$L25:

```

```

        lw      $v0,48($fp)
        lw      $v1,144($fp)
        slt     $v0,$v0,$v1
        bne     $v0,$zero,$L28
        b       $L23
$L28:
        l.d     $f2,184($fp)
        l.d     $f0,$LC0
        div.d   $f2,$f2,$f0
        l.d     $f0,152($fp)
        sub.d   $f4,$f0,$f2
        l.s     $f0,144($fp)
        cvt.d.w $f2,$f0
        l.d     $f0,184($fp)
        div.d   $f2,$f0,$f2
        l.d     $f0,$LC0
        div.d   $f0,$f2,$f0
        add.d   $f4,$f4,$f0
        l.s     $f0,144($fp)
        cvt.d.w $f2,$f0
        l.d     $f0,184($fp)
        div.d   $f2,$f0,$f2
        l.s     $f0,48($fp)
        cvt.d.w $f0,$f0
        mul.d   $f0,$f2,$f0
        add.d   $f0,$f4,$f0
        s.d     $f0,80($fp)
        l.d     $f2,192($fp)
        l.d     $f0,$LC0
        div.d   $f2,$f2,$f0
        l.d     $f0,160($fp)
        add.d   $f4,$f2,$f0
        l.s     $f0,148($fp)
        cvt.d.w $f2,$f0
        l.d     $f0,192($fp)
        div.d   $f2,$f0,$f2
        l.d     $f0,$LC0
        div.d   $f0,$f2,$f0
        sub.d   $f4,$f4,$f0
        l.s     $f0,148($fp)
        cvt.d.w $f2,$f0
        l.d     $f0,192($fp)
        div.d   $f2,$f0,$f2
        l.s     $f0,52($fp)
        cvt.d.w $f0,$f0
        mul.d   $f0,$f2,$f0
        sub.d   $f0,$f4,$f0
        s.d     $f0,88($fp)
        sw      $zero,64($fp)
        sw      $zero,68($fp)
        sw      $zero,72($fp)
        sw      $zero,76($fp)
        sw      $zero,56($fp)
$L29:
        lw      $v0,204($fp)
        addu    $v1,$v0,-1
        lw      $v0,56($fp)
        slt     $v0,$v0,$v1
        bne     $v0,$zero,$L32
        b       $L30
$L32:
        addu    $s0,$fp,64
        addu    $v1,$fp,96
        lw      $v0,88($fp)
        sw      $v0,16($sp)
        lw      $v0,92($fp)
        sw      $v0,20($sp)
        lw      $a2,80($fp)
        lw      $a3,84($fp)
        move    $a0,$v1
        la      $t9,sqrComplex
        jal     $ra,$t9
        lw      $v0,168($fp)
        sw      $v0,24($sp)

```

```

lw      $v0,172($fp)
sw      $v0,28($sp)
lw      $v0,176($fp)
sw      $v0,32($sp)
lw      $v0,180($fp)
sw      $v0,36($sp)
lw      $v0,104($fp)
sw      $v0,16($sp)
lw      $v0,108($fp)
sw      $v0,20($sp)
lw      $a2,96($fp)
lw      $a3,100($fp)
move    $a0,$s0
la      $t9,addComplexNumbers
jal     $ra,$t9
lw      $v0,64($fp)
sw      $v0,80($fp)
lw      $v0,68($fp)
sw      $v0,84($fp)
lw      $v0,72($fp)
sw      $v0,88($fp)
lw      $v0,76($fp)
sw      $v0,92($fp)
lw      $a0,64($fp)
lw      $a1,68($fp)
lw      $a2,72($fp)
lw      $a3,76($fp)
la      $t9,absComplex
jal     $ra,$t9
mov.d   $f2,$f0
l.d     $f0,$LC0
c.lt.d  $f0,$f2
bc1t    $L30
lw      $v0,56($fp)
addu    $v0,$v0,1
sw      $v0,56($fp)
lw      $v0,56($fp)
addu    $v0,$v0,1
sw      $v0,56($fp)
b       $L29

$L30:
lw      $a0,48($fp)
lw      $v0,116($fp)
sll     $v0,$v0,2
addu    $v1,$v0,4
lw      $v0,52($fp)
mult    $v1,$v0
mflo    $v1
move    $v0,$a0
sll     $v0,$v0,2
lw      $a0,120($fp)
addu    $v0,$v0,$a0
addu    $v1,$v0,$v1
lw      $v0,56($fp)
sw      $v0,0($v1)
lw      $v0,48($fp)
addu    $v0,$v0,1
sw      $v0,48($fp)
b       $L25

$L23:
lw      $v0,52($fp)
addu    $v0,$v0,1
sw      $v0,52($fp)
b       $L21

$L22:
lw      $a0,200($fp)
la      $a1,$LC1
la      $t9,fprintf
jal     $ra,$t9
lw      $a0,200($fp)
la      $a1,$LC2
lw      $a2,144($fp)
lw      $a3,148($fp)
la      $t9,fprintf

```

```

        jal      $ra,$t9
        lw       $a0,200($fp)
        la       $a1,$LC3
        lw       $a2,204($fp)
        la       $t9,fprintf
        jal      $ra,$t9
        sw       $zero,52($fp)
$L35:
        lw       $v0,52($fp)
        lw       $v1,148($fp)
        slt      $v0,$v0,$v1
        bne      $v0,$zero,$L37
        b        $L36
$L37:
        sw       $zero,48($fp)
$L38:
        lw       $v0,48($fp)
        lw       $v1,144($fp)
        slt      $v0,$v0,$v1
        bne      $v0,$zero,$L40
        b        $L39
$L40:
        lw       $a0,48($fp)
        lw       $v0,116($fp)
        sll      $v0,$v0,2
        addu     $v1,$v0,4
        lw       $v0,52($fp)
        mult     $v1,$v0
        mflo     $v1
        move     $v0,$a0
        sll      $v0,$v0,2
        lw       $a0,120($fp)
        addu     $v0,$v0,$a0
        addu     $v0,$v0,$v1
        lw       $a0,200($fp)
        la       $a1,$LC4
        lw       $a2,0($v0)
        la       $t9,fprintf
        jal      $ra,$t9
        lw       $v0,48($fp)
        addu     $v0,$v0,1
        sw       $v0,48($fp)
        b        $L38
$L39:
        lw       $a0,200($fp)
        la       $a1,$LC5
        la       $t9,fprintf
        jal      $ra,$t9
        lw       $v0,52($fp)
        addu     $v0,$v0,1
        sw       $v0,52($fp)
        b        $L35
$L36:
        lw       $a0,200($fp)
        la       $t9,fclose
        jal      $ra,$t9
        lw       $sp,112($fp)
        move     $sp,$fp
        lw       $ra,140($sp)
        lw       $fp,136($sp)
        lw       $s0,128($sp)
        addu     $sp,$sp,144
        j        $ra
        .end     processImage
        .size    processImage,.-processImage
        .rdata
        .align   2
$L36:
        .ascii   "x+-i\000"
        .align   2
$L310:
        .ascii   "-V\000"
        .align   2
$L311:

```

```

        .ascii  "--version\000"
        .align  2
$LC12:
        .ascii  "TP0 Organizacion de Computadoras version \"1.0.0\"      "
        .ascii  "                               \n\n"
        .ascii  "Integrantes:\n"
        .ascii  "  Fabrizio Cozza\n"
        .ascii  "  Kevin Cajachu\303\241n\n"
        .ascii  "  Luciano Giannotti\n\000"
        .align  2
$LC13:
        .ascii  "-h\000"
        .align  2
$LC14:
        .ascii  "--help\000"
        .align  2
$LC15:
        .ascii  "Uso:\n"
        .ascii  "  tp0 -h\n"
        .ascii  "  tp0 -V\n"
        .ascii  "  tp0 [options]\n"
        .ascii  "Opciones:\n"
        .ascii  "  -V, --version      Version del programa.\n"
        .ascii  "  -h, --help         Informacion acerca de los comandos.\n"
        .ascii  "  -r, --resolution   Cambiar resolucion de la imagen.\n"
        .ascii  "  -c, --center       Coordinadas correspondientes al punt"
        .ascii  "o central.\n"
        .ascii  "  -w, --width        Especifica el ancho de la regi\303\263"
        .ascii  "n del plano complejo por dibujar.\n"
        .ascii  "  -H, --height       Especifica el alto de la regi\303\263"
        .ascii  "n del plano complejo por dibujar.\n"
        .ascii  "  -s, --seed         Configurar el valor complejo de la s"
        .ascii  "emilla usada para generar el fractal.\n"
        .ascii  "  -o, --output       Colocar la imagen de salida.\n"
        .ascii  "Ejemplos:\n"
        .ascii  "  tp0 -o uno.pgm\n\000"
        .align  2
$LC16:
        .ascii  "-r\000"
        .align  2
$LC17:
        .ascii  "--resolution\000"
        .align  2
$LC18:
        .ascii  "Error: valor de resolucion ingresado no valido\n\000"
        .align  2
$LC19:
        .ascii  "-c\000"
        .align  2
$LC20:
        .ascii  "--center\000"
        .align  2
$LC21:
        .ascii  "Error: valor de centro ingresado no valido\n\000"
        .align  2
$LC22:
        .ascii  "-w\000"
        .align  2
$LC23:
        .ascii  "--width\000"
        .align  2
$LC24:
        .ascii  "Error: valor de ancho ingresado no valido\n\000"
        .align  2
$LC25:
        .ascii  "-H\000"
        .align  2
$LC26:
        .ascii  "--height\000"
        .align  2
$LC27:
        .ascii  "Error: valor de altura ingresado no valido\n\000"
        .align  2
$LC28:

```



```

        .ascii  "-s\000"
        .align  2
$LC29:
        .ascii  "--seed\000"
        .align  2
$LC30:
        .ascii  "Error: valor de seed ingresado no valido\n\000"
        .align  2
$LC31:
        .ascii  "-o\000"
        .align  2
$LC32:
        .ascii  "--output\000"
        .align  2
$LC33:
        .ascii  "-\000"
        .align  2
$LC34:
        .ascii  "w\000"
        .align  2
$LC35:
        .ascii  "No se puede abrir el archivo file %s!\n\000"
        .align  2
$LC36:
        .ascii  "La imagen no se pudo procesar, por favor revise los valo"
        .ascii  "res ingresados\n\000"
        .align  2
$LC37:
        .ascii  "Valores ingresados de resolucioin invalidos\n\000"
        .align  3
$LC7:
        .word   0
        .word   1073741824
        .align  3
$LC8:
        .word   138464867
        .word   -1075363142
        .align  3
$LC9:
        .word   351303579
        .word   1070083444
        .text
        .align  2
        .globl  main
        .ent    main

main:
        .frame  $fp,216,$ra          # vars= 120, regs= 3/1, args= 64,
                extra= 8
        .mask   0xd0000000,-16
        .fmask  0x00300000,-8
        .set    noreorder
        .cplod  $t9
        .set    reorder
        subu    $sp,$sp,216
        .cprestore 64
        sw      $ra,200($sp)
        sw      $fp,196($sp)
        sw      $gp,192($sp)
        s.d     $f20,208($sp)
        move    $fp,$sp
        sw      $a0,216($fp)
        sw      $a1,220($fp)
        sw      $zero,72($fp)
        lw      $v0,$LC6
        sw      $v0,136($fp)
        li      $v0,640                # 0x280
        sw      $v0,76($fp)
        li      $v0,480                # 0x1e0
        sw      $v0,80($fp)
        sw      $zero,88($fp)
        sw      $zero,92($fp)
        sw      $zero,96($fp)
        sw      $zero,100($fp)
        l.d     $f0,$LC7

```

```

s.d    $f0,104($fp)
l.d    $f0,$LC7
s.d    $f0,112($fp)
l.d    $f0,$LC8
s.d    $f0,120($fp)
l.d    $f0,$LC9
s.d    $f0,128($fp)
li     $v0,1          # 0x1
sw     $v0,148($fp)

$L42:
lw     $v0,148($fp)
lw     $v1,216($fp)
slt    $v0,$v0,$v1
bne    $v0,$zero,$L45
b      $L43

$L45:
lw     $v0,148($fp)
sll    $v1,$v0,2
lw     $v0,220($fp)
addu   $v0,$v1,$v0
lw     $a0,0($v0)
la     $a1,$LC10
la     $t9,strcmp
jal    $ra,$t9
beq    $v0,$zero,$L47
lw     $v0,148($fp)
sll    $v1,$v0,2
lw     $v0,220($fp)
addu   $v0,$v1,$v0
lw     $a0,0($v0)
la     $a1,$LC11
la     $t9,strcmp
jal    $ra,$t9
bne    $v0,$zero,$L46

$L47:
la     $a0,$LC12
la     $t9,printf
jal    $ra,$t9
sw     $zero,164($fp)
b      $L41

$L46:
lw     $v0,148($fp)
sll    $v1,$v0,2
lw     $v0,220($fp)
addu   $v0,$v1,$v0
lw     $a0,0($v0)
la     $a1,$LC13
la     $t9,strcmp
jal    $ra,$t9
beq    $v0,$zero,$L49
lw     $v0,148($fp)
sll    $v1,$v0,2
lw     $v0,220($fp)
addu   $v0,$v1,$v0
lw     $a0,0($v0)
la     $a1,$LC14
la     $t9,strcmp
jal    $ra,$t9
bne    $v0,$zero,$L48

$L49:
la     $a0,$LC15
la     $t9,printf
jal    $ra,$t9
sw     $zero,164($fp)
b      $L41

$L48:
lw     $v0,148($fp)
sll    $v1,$v0,2
lw     $v0,220($fp)
addu   $v0,$v1,$v0
lw     $a0,0($v0)
la     $a1,$LC16
la     $t9,strcmp
jal    $ra,$t9

```

```

        beq      $v0,$zero,$L51
        lw       $v0,148($fp)
        sll      $v1,$v0,2
        lw       $v0,220($fp)
        addu     $v0,$v1,$v0
        lw       $a0,0($v0)
        la       $a1,$LC17
        la       $t9,strcmp
        jal      $ra,$t9
        bne      $v0,$zero,$L50
$L51:
        lw       $v0,148($fp)
        sll      $v1,$v0,2
        lw       $v0,220($fp)
        addu     $v0,$v1,$v0
        addu     $v0,$v0,4
        lw       $v0,0($v0)
        bne      $v0,$zero,$L52
        la       $a0,$LC18
        la       $t9,printf
        jal      $ra,$t9
        li       $v0,-1
        sw       $v0,164($fp)
        b        $L41
        # 0xfffffffffffffffff

$L52:
        lw       $v0,148($fp)
        sll      $v1,$v0,2
        lw       $v0,220($fp)
        addu     $v0,$v1,$v0
        addu     $v0,$v0,4
        addu     $v1,$fp,136
        lw       $a0,0($v0)
        move     $a1,$v1
        la       $t9,strtok
        jal      $ra,$t9
        sw       $v0,144($fp)
        lw       $v0,144($fp)
        beq      $v0,$zero,$L54
        lw       $a0,144($fp)
        la       $t9,atof
        jal      $ra,$t9
        trunc.w.d $f0,$f0,$v0
        s.s      $f0,76($fp)
        lw       $v0,76($fp)
        bgtz     $v0,$L56
        li       $v0,-2
        sw       $v0,72($fp)
        b        $L56
        # 0xfffffffffffffffe

$L54:
        li       $v0,-1
        sw       $v0,72($fp)
        # 0xfffffffffffffffff

$L56:
        addu     $v0,$fp,136
        move     $a0,$zero
        move     $a1,$v0
        la       $t9,strtok
        jal      $ra,$t9
        sw       $v0,144($fp)
        lw       $v0,144($fp)
        beq      $v0,$zero,$L57
        lw       $a0,144($fp)
        la       $t9,atof
        jal      $ra,$t9
        trunc.w.d $f0,$f0,$v0
        s.s      $f0,80($fp)
        lw       $v0,80($fp)
        bgtz     $v0,$L50
        li       $v0,-2
        sw       $v0,72($fp)
        b        $L50
        # 0xfffffffffffffffe

$L57:
        li       $v0,-1
        sw       $v0,72($fp)
        # 0xfffffffffffffffff

$L50:

```

```

        lw      $v0,148($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC19
        la      $t9,strcmp
        jal     $ra,$t9
        beq     $v0,$zero,$L61
        lw      $v0,148($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC20
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L60
$L61:
        lw      $v0,148($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $v0,0($v0)
        bne     $v0,$zero,$L62
        la      $a0,$LC21
        la      $t9,printf
        jal     $ra,$t9
        li      $v1,-1
        sw      $v1,164($fp)
        b       $L41
        # 0xfffffffffffffffff
$L62:
        lw      $v0,148($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $t9,strdup
        jal     $ra,$t9
        sw      $v0,152($fp)
        lw      $v0,152($fp)
        bne     $v0,$zero,$L64
        li      $v0,-1
        sw      $v0,72($fp)
        # 0xfffffffffffffffff
$L64:
        li      $v0,1
        sw      $v0,156($fp)
        lw      $v0,152($fp)
        lb      $v1,0($v0)
        li      $v0,45
        bne     $v1,$v0,$L65
        li      $v0,-1
        sw      $v0,156($fp)
        # 0xfffffffffffffffff
$L65:
        lw      $v0,148($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        addu    $v1,$fp,136
        lw      $a0,0($v0)
        move    $a1,$v1
        la      $t9,strtok
        jal     $ra,$t9
        sw      $v0,144($fp)
        lw      $v0,144($fp)
        beq     $v0,$zero,$L66
        l.s     $f0,156($fp)
        cvt.d.w $f20,$f0
        lw      $a0,144($fp)
        la      $t9,atof
        jal     $ra,$t9

```

```

mul.d    $f0,$f20,$f0
s.d      $f0,88($fp)
lw       $a0,144($fp)
la       $t9,strlen
jal      $ra,$t9
sw       $v0,160($fp)
lw       $v1,156($fp)
li       $v0,-1                # 0xffffffffffffffff
bne      $v1,$v0,$L67
lw       $v1,152($fp)
lw       $v0,160($fp)
addu     $v0,$v1,$v0
addu     $v0,$v0,1
lb       $v1,0($v0)
li       $v0,45                # 0x2d
bne      $v1,$v0,$L68
li       $v0,-1                # 0xffffffffffffffff
sw       $v0,168($fp)
b        $L69

$L68:
li       $v1,1                # 0x1
sw       $v1,168($fp)

$L69:
lw       $v0,168($fp)
sw       $v0,156($fp)
b        $L73

$L67:
lw       $v1,152($fp)
lw       $v0,160($fp)
addu     $v0,$v1,$v0
lb       $v1,0($v0)
li       $v0,45                # 0x2d
bne      $v1,$v0,$L71
li       $v1,-1                # 0xffffffffffffffff
sw       $v1,172($fp)
b        $L72

$L71:
li       $v0,1                # 0x1
sw       $v0,172($fp)

$L72:
lw       $v1,172($fp)
sw       $v1,156($fp)
b        $L73

$L66:
li       $v0,-1                # 0xffffffffffffffff
sw       $v0,72($fp)

$L73:
addu     $v0,$fp,136
move     $a0,$zero
move     $a1,$v0
la       $t9,strtok
jal      $ra,$t9
sw       $v0,144($fp)
lw       $v0,144($fp)
beq      $v0,$zero,$L74
l.s      $f0,156($fp)
cvt.d.w  $f20,$f0
lw       $a0,144($fp)
la       $t9,atof
jal      $ra,$t9
mul.d    $f0,$f20,$f0
s.d      $f0,96($fp)
b        $L75

$L74:
li       $v0,-1                # 0xffffffffffffffff
sw       $v0,72($fp)

$L75:
lw       $a0,152($fp)
la       $t9,free
jal      $ra,$t9

$L60:
lw       $v0,148($fp)
sll      $v1,$v0,2
lw       $v0,220($fp)

```

```

        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC22
        la      $t9,strcmp
        jal     $ra,$t9
        beq     $v0,$zero,$L77
        lw      $v0,148($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC23
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L76
$L77:
        lw      $v0,148($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $v0,0($v0)
        bne     $v0,$zero,$L78
        la      $a0,$LC24
        la      $t9,printf
        jal     $ra,$t9
        li      $v0,-1
        sw      $v0,164($fp)
        b       $L41
        # 0xffffffffffffffff
$L78:
        lw      $v0,148($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $t9,atof
        jal     $ra,$t9
        s.d     $f0,104($fp)
$L76:
        lw      $v0,148($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC25
        la      $t9,strcmp
        jal     $ra,$t9
        beq     $v0,$zero,$L81
        lw      $v0,148($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC26
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L80
$L81:
        lw      $v0,148($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $v0,0($v0)
        bne     $v0,$zero,$L82
        la      $a0,$LC27
        la      $t9,printf
        jal     $ra,$t9
        li      $v1,-1
        sw      $v1,164($fp)
        b       $L41
        # 0xffffffffffffffff
$L82:
        lw      $v0,148($fp)

```

```

sll    $v1,$v0,2
lw     $v0,220($fp)
addu   $v0,$v1,$v0
addu   $v0,$v0,4
lw     $a0,0($v0)
la     $t9,atof
jal    $ra,$t9
s.d    $f0,112($fp)

$L80:
lw     $v0,148($fp)
sll    $v1,$v0,2
lw     $v0,220($fp)
addu   $v0,$v1,$v0
lw     $a0,0($v0)
la     $a1,$LC28
la     $t9,strcmp
jal    $ra,$t9
beq    $v0,$zero,$L85
lw     $v0,148($fp)
sll    $v1,$v0,2
lw     $v0,220($fp)
addu   $v0,$v1,$v0
lw     $a0,0($v0)
la     $a1,$LC29
la     $t9,strcmp
jal    $ra,$t9
bne    $v0,$zero,$L84

$L85:
lw     $v0,148($fp)
sll    $v1,$v0,2
lw     $v0,220($fp)
addu   $v0,$v1,$v0
addu   $v0,$v0,4
lw     $v0,0($v0)
bne    $v0,$zero,$L86
la     $a0,$LC30
la     $t9,printf
jal    $ra,$t9
li     $v0,-1                # 0xffffffffffffffff
sw     $v0,164($fp)
b      $L41

$L86:
lw     $v0,148($fp)
sll    $v1,$v0,2
lw     $v0,220($fp)
addu   $v0,$v1,$v0
addu   $v0,$v0,4
lw     $a0,0($v0)
la     $t9,strdup
jal    $ra,$t9
sw     $v0,160($fp)
lw     $v0,160($fp)
bne    $v0,$zero,$L88
li     $v0,-1                # 0xffffffffffffffff
sw     $v0,72($fp)

$L88:
li     $v0,1                 # 0x1
sw     $v0,156($fp)
lw     $v0,160($fp)
lb     $v1,0($v0)
li     $v0,45                # 0x2d
bne    $v1,$v0,$L89
li     $v0,-1                # 0xffffffffffffffff
sw     $v0,156($fp)

$L89:
lw     $v0,148($fp)
sll    $v1,$v0,2
lw     $v0,220($fp)
addu   $v0,$v1,$v0
addu   $v0,$v0,4
addu   $v1,$fp,136
lw     $a0,0($v0)
move   $a1,$v1
la     $t9,strtok

```

```

jal      $ra,$t9
sw       $v0,144($fp)
lw       $v0,144($fp)
beq      $v0,$zero,$L90
l.s      $f0,156($fp)
cvt.d.w  $f20,$f0
lw       $a0,144($fp)
la       $t9,atof
jal      $ra,$t9
mul.d    $f0,$f20,$f0
s.d      $f0,120($fp)
lw       $a0,144($fp)
la       $t9,strlen
jal      $ra,$t9
sw       $v0,152($fp)
lw       $v1,156($fp)
li       $v0,-1                # 0xffffffffffffffff
bne      $v1,$v0,$L91
lw       $v1,160($fp)
lw       $v0,152($fp)
addu     $v0,$v1,$v0
addu     $v0,$v0,1
lb       $v1,0($v0)
li       $v0,45                # 0x2d
bne      $v1,$v0,$L92
li       $v1,-1                # 0xffffffffffffffff
sw       $v1,176($fp)
b        $L93

$L92:
li       $v0,1                 # 0x1
sw       $v0,176($fp)

$L93:
lw       $v1,176($fp)
sw       $v1,156($fp)
b        $L97

$L91:
lw       $v1,160($fp)
lw       $v0,152($fp)
addu     $v0,$v1,$v0
lb       $v1,0($v0)
li       $v0,45                # 0x2d
bne      $v1,$v0,$L95
li       $v0,-1                # 0xffffffffffffffff
sw       $v0,180($fp)
b        $L96

$L95:
li       $v1,1                 # 0x1
sw       $v1,180($fp)

$L96:
lw       $v0,180($fp)
sw       $v0,156($fp)
b        $L97

$L90:
li       $v0,-1                # 0xffffffffffffffff
sw       $v0,72($fp)

$L97:
addu     $v0,$fp,136
move     $a0,$zero
move     $a1,$v0
la       $t9,strtok
jal      $ra,$t9
sw       $v0,144($fp)
lw       $v0,144($fp)
beq      $v0,$zero,$L98
l.s      $f0,156($fp)
cvt.d.w  $f20,$f0
lw       $a0,144($fp)
la       $t9,atof
jal      $ra,$t9
mul.d    $f0,$f20,$f0
s.d      $f0,128($fp)
b        $L99

$L98:
li       $v0,-1                # 0xffffffffffffffff

```



```

        sw        $v0,72($fp)
$L99:   lw        $a0,160($fp)
        la        $t9,free
        jal       $ra,$t9
$L84:   lw        $v0,148($fp)
        sll       $v1,$v0,2
        lw        $v0,220($fp)
        addu      $v0,$v1,$v0
        lw        $a0,0($v0)
        la        $a1,$LC31
        la        $t9,strcmp
        jal       $ra,$t9
        beq       $v0,$zero,$L101
        lw        $v0,148($fp)
        sll       $v1,$v0,2
        lw        $v0,220($fp)
        addu      $v0,$v1,$v0
        lw        $a0,0($v0)
        la        $a1,$LC32
        la        $t9,strcmp
        jal       $ra,$t9
        bne       $v0,$zero,$L44
$L101:  li        $v0,500                # 0x1f4
        sw        $v0,156($fp)
        lw        $v0,148($fp)
        sll       $v1,$v0,2
        lw        $v0,220($fp)
        addu      $v0,$v1,$v0
        addu      $v0,$v0,4
        lw        $a0,0($v0)
        la        $a1,$LC33
        la        $t9,strcmp
        jal       $ra,$t9
        bne       $v0,$zero,$L102
        la        $v0,__$SF+88
        sw        $v0,160($fp)
        b         $L103
$L102:  lw        $v0,148($fp)
        sll       $v1,$v0,2
        lw        $v0,220($fp)
        addu      $v0,$v1,$v0
        addu      $v0,$v0,4
        lw        $a0,0($v0)
        la        $a1,$LC34
        la        $t9,fopen
        jal       $ra,$t9
        sw        $v0,160($fp)
        lw        $v0,160($fp)
        bne       $v0,$zero,$L103
        lw        $v0,148($fp)
        sll       $v1,$v0,2
        lw        $v0,220($fp)
        addu      $v0,$v1,$v0
        addu      $v0,$v0,4
        la        $a0,__$SF+176
        la        $a1,$LC35
        lw        $a2,0($v0)
        la        $t9,fprintf
        jal       $ra,$t9
        li        $v1,-1                # 0xffffffffffffffff
        sw        $v1,164($fp)
        b         $L41
$L103:  lw        $v0,72($fp)
        bne       $v0,$zero,$L44
        lw        $v0,120($fp)
        sw        $v0,24($sp)
        lw        $v0,124($fp)
        sw        $v0,28($sp)
        lw        $v0,128($fp)

```

```

        sw      $v0,32($sp)
        lw      $v0,132($fp)
        sw      $v0,36($sp)
        l.d     $f0,104($fp)
        s.d     $f0,40($sp)
        l.d     $f0,112($fp)
        s.d     $f0,48($sp)
        lw      $v0,160($fp)
        sw      $v0,56($sp)
        lw      $v0,156($fp)
        sw      $v0,60($sp)
        lw      $v0,96($fp)
        sw      $v0,16($sp)
        lw      $v0,100($fp)
        sw      $v0,20($sp)
        lw      $a2,88($fp)
        lw      $a3,92($fp)
        lw      $a0,76($fp)
        lw      $a1,80($fp)
        la      $t9,processImage
        jal     $ra,$t9
        sw      $v0,72($fp)

$L44:
        lw      $v0,148($fp)
        addu    $v0,$v0,1
        sw      $v0,148($fp)
        b       $L42

$L43:
        lw      $v0,72($fp)
        sw      $v0,184($fp)
        li      $v0,-1                # 0xffffffffffffffff
        lw      $v1,184($fp)
        beq     $v1,$v0,$L108
        lw      $v1,184($fp)
        slt     $v0,$v1,0
        beq     $v0,$zero,$L112
        li      $v0,-2                # 0xfffffffffffffffe
        lw      $v1,184($fp)
        beq     $v1,$v0,$L109
        b       $L106

$L112:
        lw      $v0,184($fp)
        beq     $v0,$zero,$L107
        b       $L106

$L107:
        lw      $v0,72($fp)
        sw      $v0,164($fp)
        b       $L41

$L108:
        la      $a0,__$SF+176
        la      $a1,$LC36
        la      $t9,fprintf
        jal     $ra,$t9
        lw      $v0,72($fp)
        sw      $v0,164($fp)
        b       $L41

$L109:
        la      $a0,__$SF+176
        la      $a1,$LC37
        la      $t9,fprintf
        jal     $ra,$t9
        lw      $v0,72($fp)
        sw      $v0,164($fp)
        b       $L41

$L106:
        sw      $zero,164($fp)

$L41:
        lw      $v0,164($fp)
        move    $sp,$fp
        lw      $ra,200($sp)
        lw      $fp,196($sp)
        l.d     $f20,208($sp)
        addu    $sp,$sp,216
        j       $ra

```

```
.end      main
.size     main, .-main
.ident    "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

## 7. Bibliografía

1. GXemul.  
<http://gavare.se/gxemul/>.
2. The NetBSD project.  
<http://www.netbsd.org/>.
3. [http://es.wikipedia.org/wiki/Conjunto\\_de\\_Julia](http://es.wikipedia.org/wiki/Conjunto_de_Julia) (Wikipedia).
4. PGM format specification.  
<http://netpbm.sourceforge.net/doc/pgm.html>.
5. Generador de fractales.  
<http://usefuljs.net/fractals/>
6. GIMP.  
<https://www.gimp.org/>